CS 138: Communication I
Topics

• Network Metrics
• Layering
• Reliability
• Congestion Control
• Routing
The Fallacies of Distributed Computing

1. *The network is reliable*
2. *Latency is zero*
3. *Bandwidth is infinite*
4. The network is secure
5. *Topology doesn't change*
6. There is one administrator
7. Transport cost is zero
8. The network is homogeneous
Performance Metrics

- Bandwidth – Number of bits/unit of time the medium can transmit
- Latency – How long for message to cross network
  - Process + Queue + Transmit + Propagation
- Throughput – Effective number of bits received/unit of time
  - e.g. 10Mbps
- Goodput - *Useful* bits received per unit of time
  - Discounts protocol overhead
- Jitter – Variation in latency
Latency

• Processing Delay
  – Per message, small, limits throughput
  – e.g. to achieve full rate at a 100Mbps link, you have a budget of 120µs/pkt:

\[
\frac{100\,Mb}{s} \times \frac{pkt}{1500B} \times \frac{B}{8b} \approx 8,333\,pkt/s
\]

• Queueing Delay
  – Highly variable, offered load vs outgoing b/w

• Transmission Delay – depends on medium
  – Size/Bandwidth

• Propagation Delay – depends on medium
  – Distance/Speed of Light
Sending Packets Across

- Transmission Delay
- Propagation Delay
- Latency
Sending Packets Across

Throughput: bits / s
Which matters most, bandwidth or delay?

- How much data can we send during one RTT?
- *E.g.*, send request, receive file

- For small transfers, latency more important, for bulk, throughput more important
Many Requirements

• Modulation, encoding, framing
• Routing
• Reliability
• Flow control
• Congestion control
• Security
• …

• How to organize all of these?
ISO OSI Reference Model

1. physical
2. data link
3. network link
4. transport
5. session
6. presentation
7. application
Internet Architecture

OSI Layer 5-7

Application

End-to-End

OSI Layer 4

Internet

Net Interface

OSI Layer 3

OSI Layer 1-2
Layers

- Application – what the users sees, e.g., HTTP
- Presentation – crypto, conversion between representations
- Session – can tie together multiple streams (e.g., audio & video)
- Transport – demultiplexes, provides reliability, flow and congestion control
- Network – sends packets, using routing
- Data Link – sends frames, handles media access
- Physical – sends individual bits
How to Place Functionality

• Don’t provide functionality on a layer that some of the users won’t need
  – E.g. security, in-order-delivery conflicts with timeliness
• Don’t provide functionality on a layer when it is insufficient
  – E.g. error correction, reliability
  – This is the “End-to-end” Principle
  – Can violate when there is a performance gain
• Do provide a functionality that can be reused
  – E.g., IP routing and forwarding is useful to many layers above
Reliable Delivery

• Several sources of errors in transmission
• Error detection can discard bad frames
• Problem: if bad packets are lost, how can we ensure reliable delivery?
  – Exactly-once semantics = at least once + at most once
At Least Once Semantics

• How can the sender know packet arrived at least once?
  – Acknowledgments + Timeout

• Stop and Wait Protocol
  – S: Send packet, wait
  – R: Receive packet, send ACK
  – S: Receive ACK, send next packet
  – S: No ACK, timeout and retransmit
(a)

(b)

(c)

(d)
Stop and Wait Problems

- Duplicate data
- Duplicate acks
- Slow (channel idle most of the time!)
- May be difficult to set the timeout value
Duplicate data: adding sequence numbers

![Diagram of sender and receiver with frames and acknowledgments]

Frame 0
ACK 0
Frame 1
ACK 1
Frame 0
ACK 0
...
At Most Once Semantics

• How to avoid duplicates?
  – Uniquely identify each packet
  – Have receiver and sender remember

• Stop and Wait: add 1 bit to the header
  – Why is it enough?
Going faster: sliding window protocol

- Still have the problem of keeping pipe full
  - Generalize approach with > 1-bit counter
  - Allow multiple outstanding (unACKed) frames
  - Upper bound on unACKed frames, called *window*
How big should the window be?

- How many bytes can we transmit in one RTT?
  - \( \text{BW B/s} \times \text{RTT s} \Rightarrow \text{“Bandwidth-Delay Product”} \)
Maximizing Throughput

- Can view network as a pipe
  - For full utilization want bytes in flight \( \geq \) bandwidth \( \times \) delay
  - But don’t want to overload the network
- What if protocol doesn’t involve bulk transfer?
  - Get throughput through concurrency – service multiple clients simultaneously
Problem: on the Internet, the pipe varies

• Different paths
• Queues along the way
• Other flows

• Q: How to set the window size then?
Efficiency

• 3 goals:
  – Utilize the network
  – Don’t overwhelm the receiver: flow control
  – Don’t overwhelm the network: congestion control
Congestion Control

• 3 Key Challenges
  – Determining the available capacity in the first place
  – Adjusting to changes in the available capacity
  – Sharing capacity between flows

• Idea
  – Each source determines network capacity for itself
  – Rate is determined by window size
  – Uses implicit feedback (drops, delay)
  – ACKs pace transmission (self-clocking)
Ack Clocking
Fast Start
Slow Start
Congestion Control (1)

1)  

2)  

3)
Congestion Control (2)

4) [Diagram of congestion control scenario 1]

5) [Diagram of congestion control scenario 2]
Dealing with Congestion

• Assume losses are due to congestion
• After a loss, reduce congestion window
  – How much to reduce?
How much to reduce window?

• Crude model of the network
  – Let \( L_i \) be the load (# pkts) in the network at time \( i \)
  – If network uncongested, roughly constant \( L_i = N \)

• What happens under congestion?
  – Some fraction \( \gamma \) of packets can’t exit the network
  – Now \( L_i = N + \gamma L_{i-1} \), or \( L_i \approx g^i L_0 \)
  – Exponential increase in congestion

• Sources must decrease offered rate exponentially
  – i.e, multiplicative decrease in window size
  – TCP chooses to cut window in half
How to use extra capacity?

• Network signals congestion, but says nothing of underutilization
  – Senders constantly try to send faster, see if it works
  – So, increase window if no losses… By how much?
• Multiplicative increase?
  – Easier to saturate the network than to recover
  – Too fast, will lead to saturation, wild fluctuations
• Additive increase?
  – Won’t saturate the network
  – Remember fairness?
Chiu Jain Phase Plots

- **Fair:** $A = B$
- **Efficient:** $A + B = C$

Goal: fair and efficient!
Chiu Jain Phase Plots

Flow Rate A vs. Flow Rate B

- **Fair**: $A = B$
- **Efficient**: $A + B = C$

Points:
- MI MD

Copyright © 2012 Thomas W. Doeppner. All rights reserved.
Chiu Jain Phase Plots

Flow Rate A vs. Flow Rate B

- Fair: \( A = B \)
- Efficient: \( A + B = C \)

AIAD point on the graph
Chiu Jain Phase Plots

- Fair: $A = B$
- Efficient: $A + B = C$

Diagram showing flow rate $A$ vs. flow rate $B$ with points indicating AIMD behavior.
AIMD Trace

- AIMD produces sawtooth pattern of window size
  - Always probing available bandwidth